

# Lógica combinatoria

## Breve descripción:

La lógica combinatoria utiliza compuertas lógicas como OR, AND y NOT para realizar funciones específicas. Representa operaciones mediante esquemas lógicos y tablas de verdad. Su simplificación optimiza diseños usando propiedades de Boole, teoremas de De Morgan y mapas de Karnaugh, reduciendo costos y espacio. Se aplica en circuitos digitales para procesos como control y supervisión.

## Tabla de contenido

Introducción .....	1
1. Lógica combinatoria.....	2
Esquema lógico o plano.....	3
Simplificación de funciones lógicas.....	4
Validación de la simplificación .....	6
Esquema lógico simplificado versus esquema lógico sin simplificar .....	7
Síntesis .....	18
Material complementario.....	19
Glosario .....	20
Referencias bibliográficas .....	21
Créditos .....	22

## Introducción

La lógica combinatoria es un componente fundamental en la electrónica digital, permitiendo integrar diferentes compuertas lógicas para resolver problemas específicos y realizar funciones complejas. Este enfoque se basa en operaciones como OR, AND y NOT, que constituyen la base para procesar y manipular señales binarias en circuitos digitales.

Una de las herramientas principales en la lógica combinatoria son los esquemas lógicos y las tablas de verdad, los cuales permiten visualizar y analizar el comportamiento de las funciones lógicas en función de todas las posibles combinaciones de entrada. Estas representaciones facilitan el diseño de circuitos eficientes y precisos en aplicaciones prácticas.

Además, el proceso de simplificación de funciones lógicas, mediante el uso de propiedades de Boole, teoremas de De Morgan y mapas de Karnaugh, optimiza los diseños reduciendo la cantidad de compuertas necesarias. Esto no solo ahorra espacio y costos, sino que también mejora el rendimiento general de los circuitos, destacando la importancia de este concepto en el ámbito tecnológico.

## 1. Lógica combinatoria

La lógica combinatoria implica la integración de diferentes compuertas en un único circuito o esquema lógico.

### Ejercicio práctico 1

Realizar el esquema lógico y la tabla de verdad a partir de la siguiente función lógica:  $(A + B) \cdot (B + C)$ .

Para construir el esquema lógico basado en la función dada, primero se identifican las operaciones involucradas para que se cumpla la función, de la siguiente forma:

$$(A + B) \cdot (B + C)$$

3 operaciones: (+, \*, +)

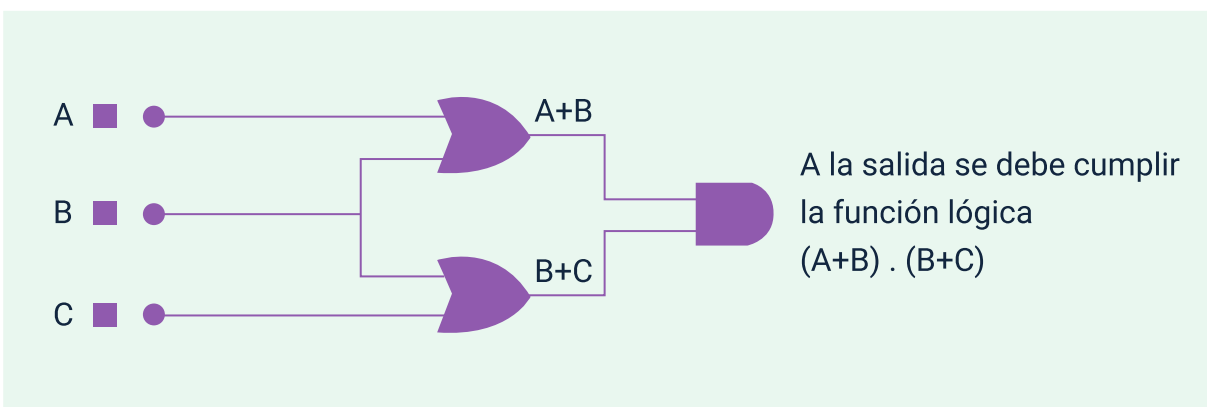
A continuación, se identifican las compuertas que corresponden a cada operación:

**Tabla 1.** Relación entre funciones lógicas, operaciones y compuertas

Función o salida	Operación	Compuerta
$A + B$	+	OR
$B + C$	+	OR
$(A + B) \cdot (B + C)$	·	AND

De esta forma, el esquema lógico quedaría diseñado considerando las compuertas correspondientes a cada operación.

**Figura 1.** Diagrama de compuertas lógicas



## Esquema lógico o plano

Antes de construir la tabla de verdad, es fundamental tener en cuenta:

**Tabla de verdad.** Es una representación gráfica de todos los valores que puede asumir la función lógica para cada una de las posibles combinaciones de las variables de entrada.

El número de combinaciones posibles se calcula como  $2^n$ , siendo  $n$  el número de variables.

### Ejemplo.

Para la función lógica  $F = (A+B) \cdot C$

$n=3$  (ya que las variables son A, B y C).

$2^3=8$  (número de combinaciones posibles y filas que debe tener la tabla).

La tabla de verdad y los niveles lógicos que cumplen dicha función en la salida son:

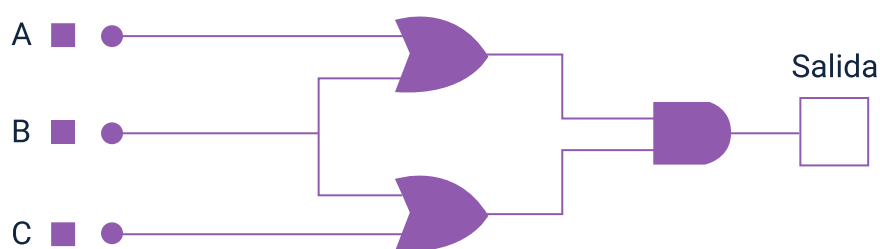
**Tabla 2.** Tabla de verdad

Variables o entradas	Variables o entradas	Variables o entradas	Salidas parciales	Salidas parciales	Función lógica
A	B	C	A + B	B + C	$(A + B) * (B + C)$
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	1	1

### Combinaciones

Para comprobar la tabla realice la simulación:

**Figura 2.** Diagrama lógico con compuertas OR y AND



### Simplificación de funciones lógicas

Al diseñar esquemas lógicos para cumplir funciones o procesos específicos, puede generarse inicialmente una función extensa que requiera un gran número de compuertas lógicas. Sin embargo, estas funciones pueden simplificarse mediante las propiedades de Boole, los teoremas de Morgan y los mapas de Karnaugh.

Como ejercicio práctico, se continuará trabajando con la función lógica del ejercicio 1.

## Ejercicio práctico 2

Simplificar la función lógica:  $(A+B) \cdot (B+C)$

Antes de simplificar la función, es importante recordar las propiedades de Boole y los teoremas de Morgan:

**Tabla 3.** Propiedades o Reglas del Álgebra de Boole

Propiedad o Regla	Ejemplo
Simplificación de negación	$\bar{\bar{A}} = A$
Simplificación de igualdad	$A + A = A$ $A \cdot A = A$
Inversa	$A + \bar{A} = 1$ $A \cdot \bar{A} = 0$
Identidad	$A + 0 = A$ $A \cdot 1 = A$ $A + 1 = 1$ $A \cdot 0 = 0$
Conmutativa	$A + B = B + A$ $A \cdot B = B \cdot A$
Asociativa	$A + (B + C) = (A + B) + C$ $A \cdot (B \cdot C) = (A \cdot B) \cdot C$
Distributiva	$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$ $(A + B) \cdot C = (A \cdot C) + (B \cdot C)$

Propiedad o Regla	Ejemplo
Más reglas de simplificación o leyes de absorción	$A + A \cdot B = A$ $A \cdot (A + B) = A$
Teorema de Morgan	$A \overline{\cdot} B = A' + B'$ $A \overline{+} B = A' \cdot B'$

El proceso de simplificación es:

- a) Reorganización de la función usando la propiedad conmutativa

$$(A + B) * (B + C) = (B + A) * (B + C)$$

- b) Aplicación de la propiedad distributiva

$$(B + A) * (B + C) = B + (A * C)$$

- c) Resultado simplificado

d)  $(A + B) * (B + C) = B + (A * C)$

## Validación de la simplificación

Para validar que la función simplificada cumple con la misma salida que la función original, se debe:

- Construir el esquema lógico simplificado y compararlo con el inicial.
- Comprobar que las tablas de verdad coincidan para ambas expresiones.
- Analizar cómo la simplificación reduce el número de compuertas necesarias.

Esto evidencia que la simplificación no solo conserva la lógica original, sino que optimiza el diseño del esquema lógico.

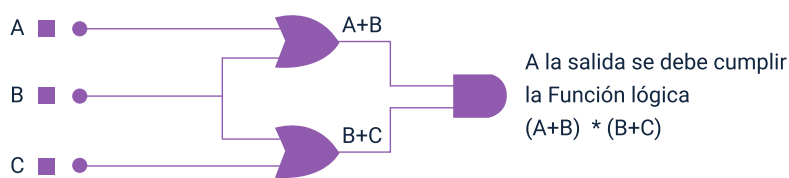
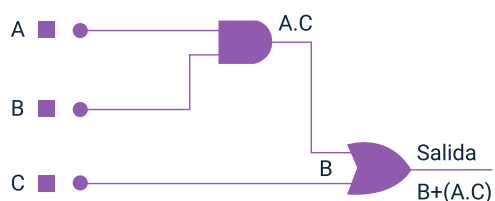
Función lógica sin simplificar:  $(A+B)*(B+C)$



Función lógica simplificada:  $B+(A*C)$

**Figura 3.** Representación de funciones lógicas con compuertas

Función ó Salida	Operación	Compuerta
A.C	•	AND
$B+(A.C)$	+	OR



### Esquema lógico simplificado versus esquema lógico sin simplificar

Ambos esquemas realizan la misma función lógica, pero el esquema simplificado utiliza menos compuertas, lo que resulta en una reducción de costos y espacio. Esto resalta la importancia de simplificar las funciones lógicas al diseñar circuitos.

A continuación, se procederá a comprobar que las dos funciones, la original  $(A+B)*(B+C)$  y la simplificada  $B+(A*C)$ , generan los mismos resultados mediante sus respectivas tablas de verdad:

**Figura 4.** Tabla comparativa de funciones lógicas sin simplificar y simplificada

Variables o entradas			Función lógica sin simplificar	Variables o entradas			Salida parcial	Función lógica simplificada
A	B	C	$(A+B) \cdot (B+C)$	A	B	C	$A \cdot C$	$A \cdot C$
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0	0
0	1	1	1	0	1	1	0	0
1	0	0	0	1	0	0	0	0
1	0	1	1	1	0	1	1	1
1	1	0	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1

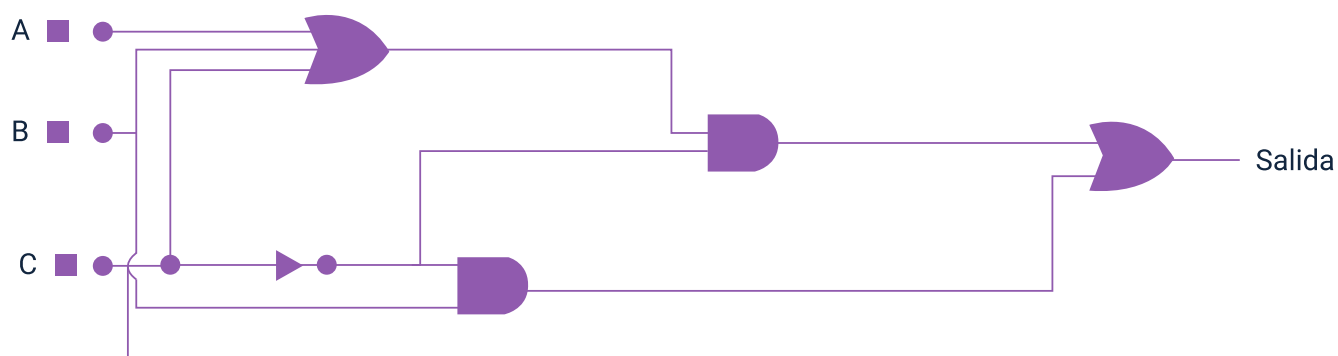
$$\boxed{\phantom{00000000}} = \boxed{\phantom{00000000}}$$

Se evidencia que las salidas para cada combinación son las mismas.

### Ejercicio práctico 3

A partir del esquema lógico, se debe elaborar la tabla de verdad y determinar la función lógica. Luego, simplificar la función lógica obtenida y comprobar que las salidas coinciden con las de la función original sin simplificar.

**Figura 5.** Circuito lógico con compuertas combinadas

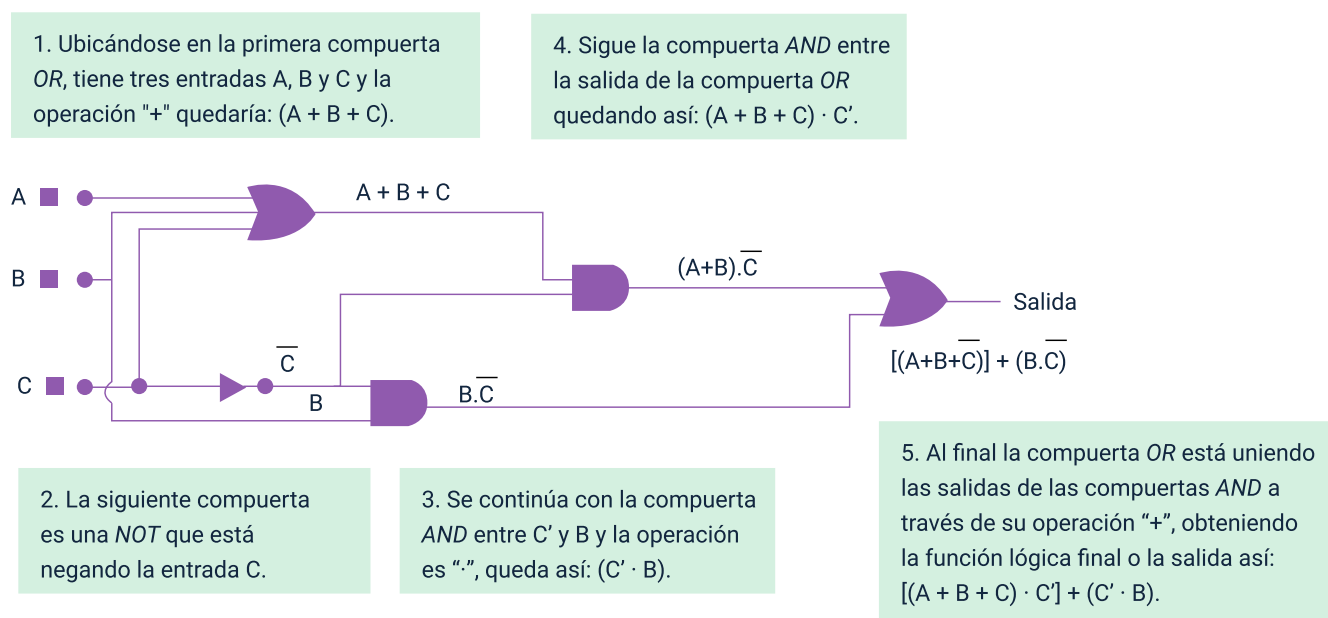


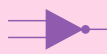
El cable de la entrada B pasa por encima del cable de la entrada C, no se cruzan

### **Solución:**

Para determinar la función lógica del esquema, es necesario identificar el tipo de compuertas lógicas y las operaciones que realiza cada una, de la siguiente manera:

**Figura 6.** Diagrama de circuito lógico y tabla de operaciones



Compuerta	Función ó Salida	Operación
2 OR		+
NOT		Inversor o negación
AND		•

Con la función lógica, se procede a construir la tabla de verdad. La tabla de verdad y los niveles lógicos que corresponden a la salida de dicha función son los siguientes:

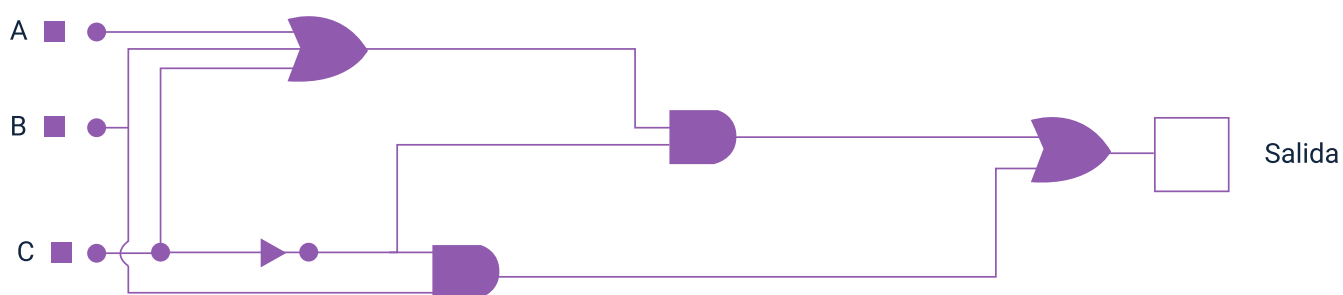
**Tabla 4.** Tabla de verdad para función lógica compuesta

Variables o entradas	Variables o entradas	Variables o entradas	Salidas parciales	Salidas parciales	Salidas parciales	Salidas parciales	Función lógica
A	B	C	$A+B+C$	$C'$	$(A+B+C) \cdot C'$	$B \cdot C'$	$[(A+B+C) \cdot C'] + B \cdot C'$

Variables o entradas	Variables o entradas	Variables o entradas	Salidas parciales	Salidas parciales	Salidas parciales	Salidas parciales	Función lógica
0	0	0	0	1	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	1	1	1	1
0	1	1	1	0	0	0	0
1	0	0	1	1	1	0	1
1	0	1	1	0	0	0	0
1	1	0	1	1	1	1	1
1	1	1	1	0	0	0	0

Para comprobar la tabla se debe realizar la simulación:

**Figura 7.** Circuito lógico con compuertas combinadas



**Simplificación de la función anterior:**

$$[(A+B+C) \cdot C'] + (B \cdot C')$$

a) Multiplicación de  $C'C'C'$  por la operación  $(A+B+C)$

$$(A \cdot C') + (B \cdot C') + (C \cdot C') + (B \cdot C')$$

b) Aplicación de la propiedad inversa ( $C \cdot C' = 0$ )

$$(A \cdot C') + (B \cdot C') + (B \cdot C')$$

c) Simplificación de igualdad ( $B \cdot C' + B \cdot C' = B \cdot C'$ )

$$(A \cdot C') + (B \cdot C')$$

d) Aplicación de la propiedad distributiva ( $C' \cdot (A+B)$ )

$$C' \cdot (A+B)$$

e) Resultado simplificado

$$[(A+B+C) \cdot C'] + (B \cdot C') = C' \cdot (A+B)$$

f) Conclusión

Función inicial = Función simplificada.

#### **Propiedades utilizadas:**

- **Propiedad inversa**

$$C \cdot C' = 0.$$

- **Propiedad de simplificación de igualdad**

$$B \cdot C' + B \cdot C' = B \cdot C'$$

- **Propiedad distributiva**

$$C' \cdot (A+B) = (A \cdot C') + (B \cdot C')$$

Ahora, se comprobará que ambas funciones generan los mismos resultados mediante sus tablas de verdad.

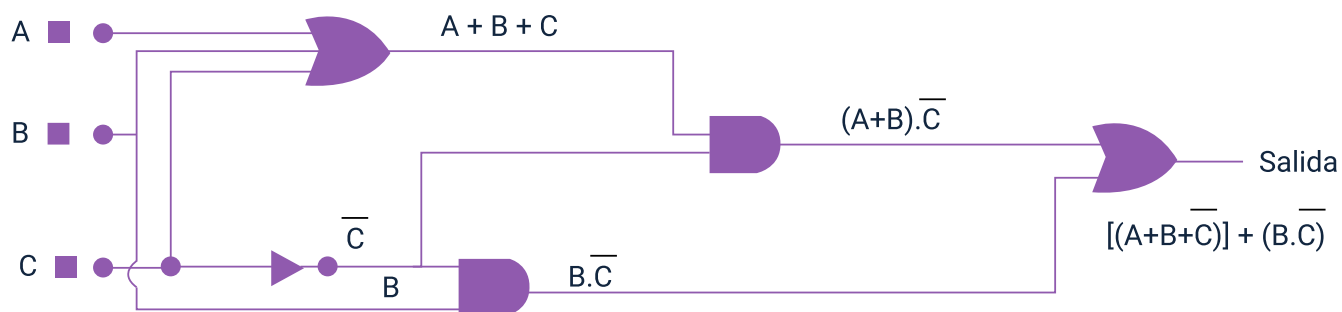
**Figura 8.** Comparación de función lógica sin simplificar y simplificada

Variables o entradas			Función lógica sin simplificar	Variables o entradas			Salida parcial		Función lógica simplificada
A	B	C	$[(A+B+C)] + B + C'$	A	B	C	A+B	C'	$C'*(A+B)$
0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1	1	1
0	1	1	0	0	1	1	1	0	0
1	0	0	1	1	0	0	1	1	1
1	0	1	0	1	0	1	1	0	0
1	1	0	0	1	1	0	1	1	1
1	1	1	0	1	1	1	1	0	0

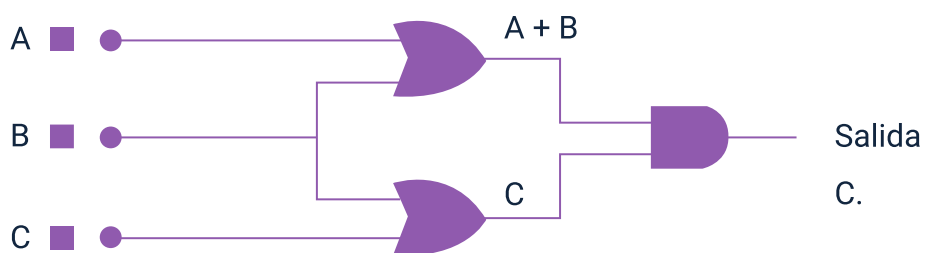
**=**

Se evidencia que las salidas para cada combinación son las mismas

**Figura 9.** Esquema lógico inicial



**Figura 10.** Esquema lógico simplificado



- **Caso**

El gerente de una empresa de transporte desea tener mayor control sobre el registro del equipaje de los pasajeros, por lo que solicita al ingeniero un circuito digital capaz de supervisar el paso de las maletas durante el recorrido de revisión.

- **Objetivo**

Diseñar un circuito digital capaz de detectar el equipaje desde que ingresa a la zona de carga, hasta que pasa la revisión y el registro, permitiendo finalmente su acceso a la zona de carga.

Las variables de entrada son:

- **Sensor Puerta 1**

Detecta el equipaje para permitir la apertura de la puerta 1 y el avance hacia el primer control. La salida es '0' cuando detecta equipaje y '1' cuando no.

- **Pulsador Puerta 2**

Permite la apertura de la puerta 2 si se oprime el botón y la puerta 1 previamente estuvo en '0'. La salida es '1' si el botón es oprimido y '0' si no.

- **Sensor Puerta 3**

Detecta el código de barras del tiquete para abrir la puerta 3 y permitir el paso a la zona de carga. La salida es '1' cuando detecta el código y '0' cuando no.

## **Solución**

Identificar entradas y salidas:





**Figura 11.** Tabla de condiciones de operación para puertas con diferentes entradas y salidas

Puerta 1 (una entrada)		Puerta 2 (dos entradas)		Puerta 3 (2 entradas)	
Entrada	Salida	Entrada	Salida	Entrada	Salida
Sensor de objetos en "0".	Salida puerta 1 en "1".	Salida puerta 1 en "1".	Se abre la puerta 2 en "1".	Salida puerta 2 en "1".	Se abre la puerta 3 en "1".
		Botón en "1".		Sensor que detecta tiquete en "1".	Se permite cargar el equipaje.

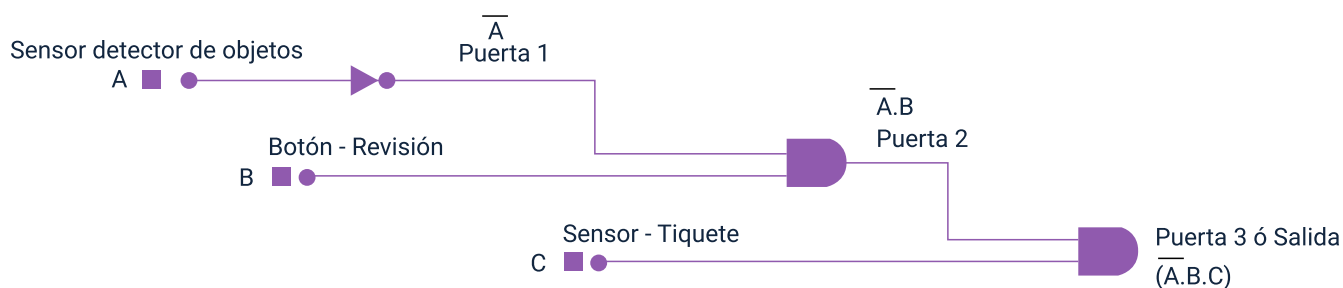
## Identificación de compuertas lógicas

**Figura 12.** Tabla de operación y compuertas lógicas para tres puertas

Puerta 1 (una entrada)		Puerta 2 ( dos entradas )		Puerta 3 ( dos entradas )	
Operación	Compuerta	Operación	Compuerta	Operación	Compuerta
Cuando ingresa "0" la salida sea "1" es decir sea negada	Inversor o Negación	Condición: cuando las salidas de la puerta 1 y el botón sean igual o "1" entonces la salida será "1"	La puerta lógica que cumple esa condición es la puerta ADN	Condición: cuando las salidas de la puerta 2 y el sensor que detecta el tiquete sean cada una igual a "1" entonces la salida será "1"	También para esta función la puerta apropiada es la AND
					

El esquema lógico, según las entradas y salidas de las compuertas, es el siguiente:

**Figura 13.** Diagrama lógico de operación para control de puertas



La función lógica es:

$$(A * B * C) = (A' * B * C)$$

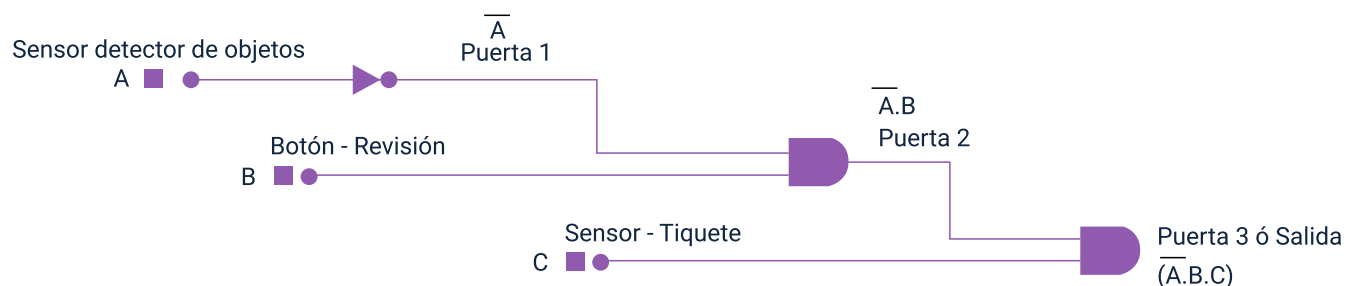
Verifique, mediante la tabla de verdad, que el circuito cumple con el proceso requerido por el gerente. Esto implica evaluar todas las combinaciones posibles de las entradas y comprobar que las salidas corresponden a las condiciones establecidas para cada puerta del sistema.

**Figura 14.** Tabla de verdad y condiciones de apertura de puertas

Variable o entradas			Puerta 1		Puerta 1		Función lógica o salida puerta 3
A	B	C	A'	Si se cumple	A' · B	Si se cumple la condición	Función lógica o salida puerta 3
0	0	0	1	Se abre	0	No abre	0
0	0	1	1	Se abre	0	No abre	0
0	1	0	1	Se abre	1	Se abre	0
0	1	1	1	Se abre	1	Se abre	1 Se abre luego si se cumple
1	0	0	0	No abre	0	No abre	0
1	0	1	0	No abre	0	No abre	0
1	1	0	0	No abre	0	No abre	0
1	1	1	0	No abre	0	No abre	0

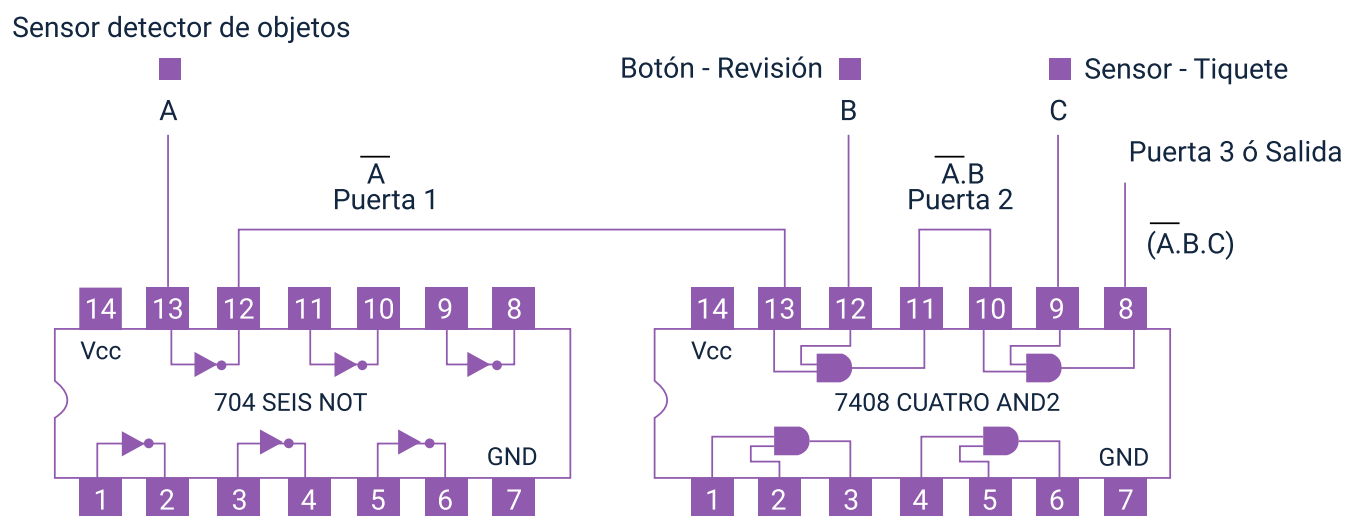
Para finalizar se identificará que tipo de CI son necesarios para realizar el circuito:

**Figura 15.** Diagrama lógico de control de puertas



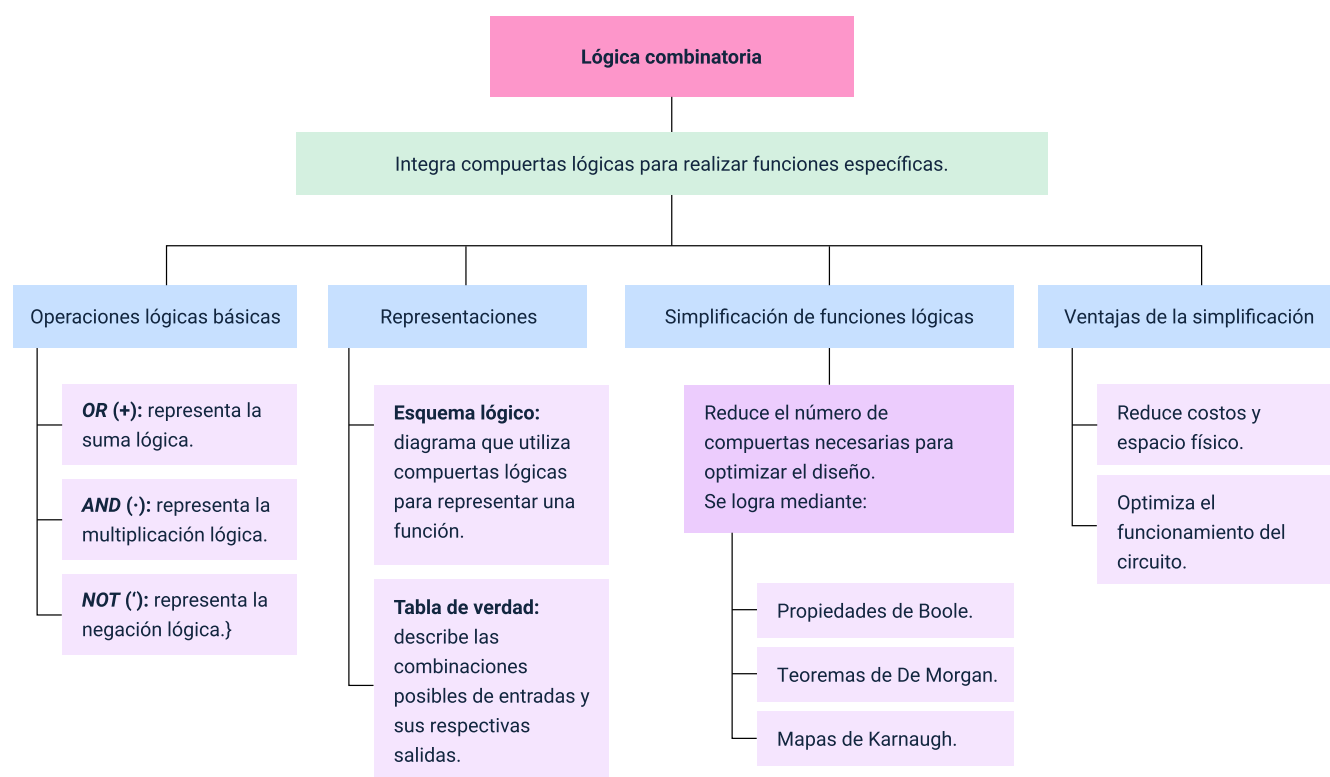
En el esquema se identifican tres compuertas necesarias para cumplir las funciones específicas del proceso, de las cuales dos son del tipo AND. Por lo tanto, los circuitos integrados adecuados para el diseño son:

**Figura 16.** Implementación del control de puertas con circuitos integrados



## Síntesis

A continuación, se presenta una síntesis de la temática estudiada en el componente formativo.



## Material complementario

Tema	Referencia	Tipo de material	Enlace del recurso
Lógica combinatoria	Fernando González. (2020). LÓGICA COMBINATORIA COMPUERTAS LÓGICAS CLASE 1. [Archivo de video] YouTube.	Video	<a href="https://www.youtube.com/watch?v=b096QGpxj38&amp;ab_channel=FernandoGonz%C3%A1lez">https://www.youtube.com/watch?v=b096QGpxj38&amp;ab_channel=FernandoGonz%C3%A1lez</a>
Lógica combinatoria	Mundo Electrónica (2020). Compuertas lógicas y lógica combinacional   Curso de electrónica digital   #5 [Archivo de video] YouTube.	Video	<a href="https://www.youtube.com/watch?v=OAA2B50e9nA&amp;ab_channel=MundoElectr%C3%B3nica">https://www.youtube.com/watch?v=OAA2B50e9nA&amp;ab_channel=MundoElectr%C3%B3nica</a>
Lógica combinatoria	Electrónica FP. (2019). DeMorgan (Ejercicio) [Archivo de video] YouTube.	Video	<a href="https://www.youtube.com/watch?v=N5YXG0KKLCc&amp;ab_channel=Electr%C3%B3nicaFP">https://www.youtube.com/watch?v=N5YXG0KKLCc&amp;ab_channel=Electr%C3%B3nicaFP</a>

## Glosario

**Compuerta lógica:** dispositivo digital que realiza operaciones básicas como and, or o not.

**Esquema lógico:** representación gráfica de funciones lógicas mediante compuertas.

**Lógica combinatoria:** rama de la electrónica digital que utiliza compuertas lógicas para realizar funciones específicas.

**Mapas de Karnaugh:** técnica gráfica para simplificar funciones lógicas en circuitos digitales.

**Operación AND:** representa la multiplicación lógica, donde la salida es 1 solo si todas las entradas son 1.

**Operación NOT:** representa la negación lógica, invirtiendo el valor de la entrada.

**Operación OR:** representa la suma lógica, donde la salida es 1 si alguna entrada es 1.

**Simplificación lógica:** proceso de optimizar funciones lógicas reduciendo el número de compuertas necesarias.

**Tabla de verdad:** herramienta para describir todas las combinaciones posibles de entradas y sus salidas.

**Teoremas de De Morgan:** reglas matemáticas que permiten simplificar expresiones lógicas mediante negaciones.

## Referencias bibliográficas

Cidead, (s.f). Material interactivo sobre Lógica Binaria.

[http://recursostic.educacion.es/secundaria/edad/4esotecnologia/quincena5/4q2\\_inde x.htm](http://recursostic.educacion.es/secundaria/edad/4esotecnologia/quincena5/4q2_inde x.htm)

Mc Graw Hill, (s.f), Introducción a los sistemas digitales. Unidad 1. En Mc Graw Hill. <http://www.mcgraw-hill.es/bcv/guide/capitulo/844817156X.pdf>

Neuroproductions, (s.f). Simulador On line.

ProfesorMolina, (s.f). Función interactiva de compuertas.

## Créditos

Nombre	Cargo	Centro de Formación y Regional
Milady Tatiana Villamil Castellanos	Responsable del ecosistema	Dirección General
Olga Constanza Bermúdez Jaimes	Responsable de línea de producción	Centro de Servicios de Salud - Regional Antioquia
Magda Melissa Rodríguez Celis	Experto temático	Centro de Desarrollo Agroempresarial - Regional Cundinamarca
Paola Alexandra Moya Peralta	Evaluadora instruccional	Centro de Servicios de Salud - Regional Antioquia
Carlos Julián Ramírez Benítez	Diseñador de contenidos digitales	Centro de Servicios de Salud - Regional Antioquia
Edwin Sneider Velandia Suárez	Desarrollador full stack	Centro de Servicios de Salud - Regional Antioquia
Jaime Hernán Tejada Llano	Validador de recursos educativos digitales	Centro de Servicios de Salud - Regional Antioquia
Margarita Marcela Medrano Gómez	Evaluador para contenidos inclusivos y accesibles	Centro de Servicios de Salud - Regional Antioquia
Daniel Ricardo Mutis Gómez	Evaluador para contenidos inclusivos y accesibles	Centro de Servicios de Salud - Regional Antioquia